

UNIVERSITÉ DE MONTPELLIER

M1 INFORMATIQUE

PARCOURS GÉNIE LOGICIEL

**Projet Programmation Mobile (HAI811I) : Application
pour la Gestion des Intérims**

Grimaud Matthieu

Nizard Alexis

Papin Adrien



Table des matières

Partie 1	Page 3
1.1 Architecture du projet	3
1.1.1 Design pattern Model-View-Controller (MVC)	3
1.1.2 Base de données Firebase (Cloud Firestore)	5
1.1.2.1 Architecture de la base de données	5
1.1.2.2 Méthodes utilisées pour travailler ensemble dessus	6
1.1.3 Utilisation d'un tableau Trello pour organiser notre travail	7
1.1.4 Outils utilisés	8
Partie 2	Page 9
2.1 Liste détaillée des fonctionnalités réalisées	9
2.1.1 Utilisateurs non connectés (anonymes)	9
2.1.2 Utilisateurs connectés en tant que Candidat	15
2.1.3 Utilisateurs connectés en tant qu'Employeur/Agence	20
Partie 3	Page 35
3.1 Conclusion	35

Introduction

Le projet consistait à développer une application mobile d'Interim pour gérer des interactions entre les chercheurs d'emploi en intérim, les employeurs, les agences d'intérim et les gestionnaires de la plateforme.

L'application d'abord permettre l'accès à des utilisateurs **anonymes**. Ces utilisateurs, sans créer de compte, peuvent consulter les offres d'emploi disponibles, utiliser les outils de recherche et de filtrage, et consulter les détails des offres. Bien qu'ils ne puissent pas postuler directement via l'application, ils peuvent tout de même bénéficier d'une vue d'ensemble des diverses offres qui seront publiées par les employeurs et des opportunités disponibles.

En ce qui concerne les chercheurs d'emploi, **candidats inscrits**, l'application doit proposer notamment des services pour faciliter le processus de recherche d'emploi. Les utilisateurs peuvent bénéficier d'un profil détaillé et utiliser des filtres pour rechercher des offres d'emploi pertinentes. Ils peuvent enregistrer des offres, postuler directement via l'application et gérer leurs candidatures en cours. L'application propose également des services d'alertes et de notifications pour informer les utilisateurs des nouvelles offres correspondant à leurs critères de recherche. Pour aider les utilisateurs dans leur candidature, l'application peut également proposer des modèles de CV et de lettres de motivation, ainsi qu'un soutien à la rédaction de ces documents. Les utilisateurs peuvent également consulter des statistiques d'emploi pour évaluer l'employabilité en fonction du métier, de la période et de la zone géographique.

Pour les **employeurs et les agences d'intérim**, l'application offre une plateforme pour gérer les offres d'emploi et interagir avec les candidats. Ils peuvent publier des offres d'emploi, consulter et gérer les offres déjà publiées, et même créer des catégories d'offres. Ils ont également la possibilité de consulter, accepter, refuser et répondre aux candidatures, et de bloquer ou signaler un candidat si nécessaire.

Enfin, pour le rôle de **gestionnaire**, ces derniers peuvent consulter les inscriptions des employeurs et des agences, valider ou refuser ces inscriptions, et contacter les employeurs ou les agences si nécessaire. Les gestionnaires ont également la possibilité de gérer les utilisateurs bloqués ou signalés. L'application fournit également aux gestionnaires la possibilité d'accéder à diverses statistiques.

Le lien de notre Github est le suivant : <https://github.com/AlexisNizard/ProjetMobile>

1.1 Architecture du projet

1.1.1 Design pattern Model-View-Controller (MVC)

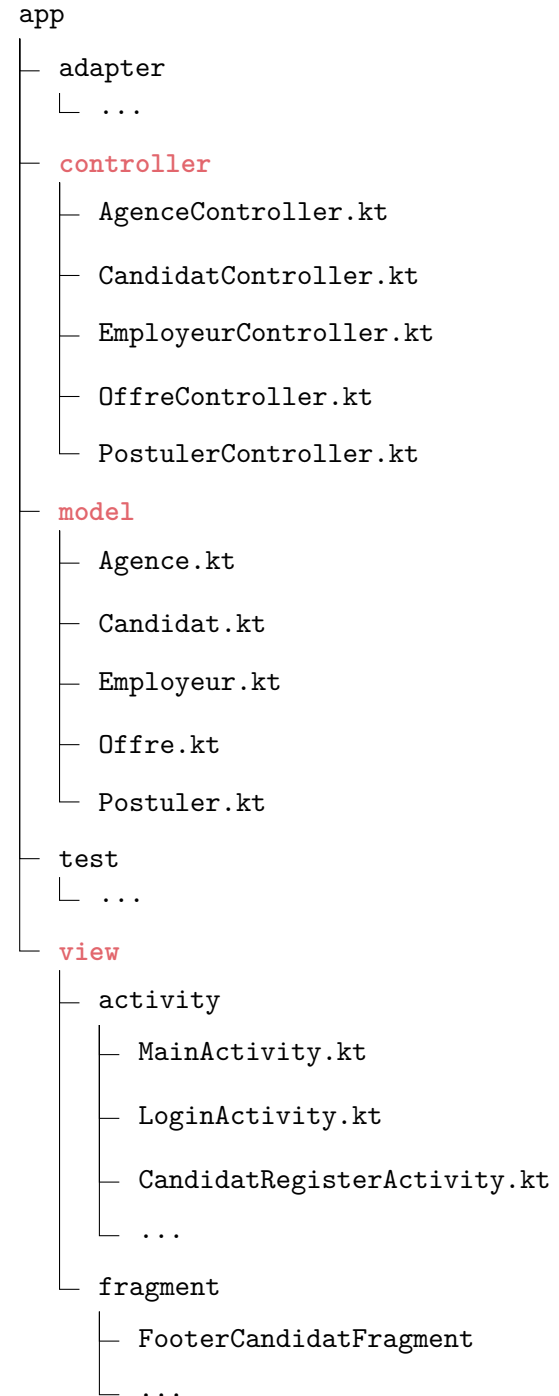
Notre projet respecte une architecture **Model-View-Controller** et est donc organisé en trois packages principaux : view, model et controller.

Le **Modèle** représente les **données de l'application**. Dans notre cas, c'est le package model qui contient toutes les données nécessaires pour notre application. Par exemple, la classe Offre qui est une représentation d'une offre d'emploi.

```
data class Offre(  
    var id: String? = null,  
    var titre: String? = null,  
    var entreprise: String? = null,  
    ...  
    var dateCreation: Timestamp? = null,  
    var nbrClick: Int = 0  
)
```

Le **Contrôleur** est chargé de la **gestion de l'interaction avec la base données**. Dans notre projet, pour la classe Offre, le controller OffreController contient les méthodes nécessaires comme la récupération d'une offre d'emploi, la mise à jour d'une offre, l'insertion d'une nouvelle offre, etc.

```
class OffreController {  
  
    private val db =  
        FirebaseFirestore.getInstance()  
    private val offresCollection =  
        db.collection("offres")  
    ...  
  
    fun getOffre(id: String):  
        Task<DocumentSnapshot> {  
        return  
            offresCollection.document(id).get()  
        }  
    fun insertOffre(offre: Offre) {  
        ...  
  
        offresCollection.add(offre)  
            .addOnSuccessListener { documentReference ->  
                ...  
            }  
            .addOnFailureListener { exception ->
```



```

        ...
    }
}
...
}

```

La **Vue** représente l'**interface utilisateur de l'application**. Dans notre cas, le package view contient les classes responsables de l'interaction avec l'utilisateur. L'activité CreerOffreEmploiActivity, par exemple, gère la création d'une nouvelle offre d'emploi. latex

```

class CreerOffreEmploiActivity : AppCompatActivity() {

    private lateinit var offreController: OffreController
    ...

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_creer_offre_emploi)

        offreController = OffreController()

        ...
        creerOffreButton.setOnClickListener {
            ...
            registerOffre()
            finish()
        }
    }

    private fun registerOffre() {
        ...
        val offre = Offre(
            ...
        )
        offreController.insertOffre(offre)
    }
}

```

L'utilisation de cette architecture MVC nous a permis de séparer efficacement les préoccupations, ce qui a facilité la maintenance du code et la collaboration au sein de l'équipe sur le projet.

1.1.2 Base de données Firebase (Cloud Firestore)

1.1.2.1 Architecture de la base de données

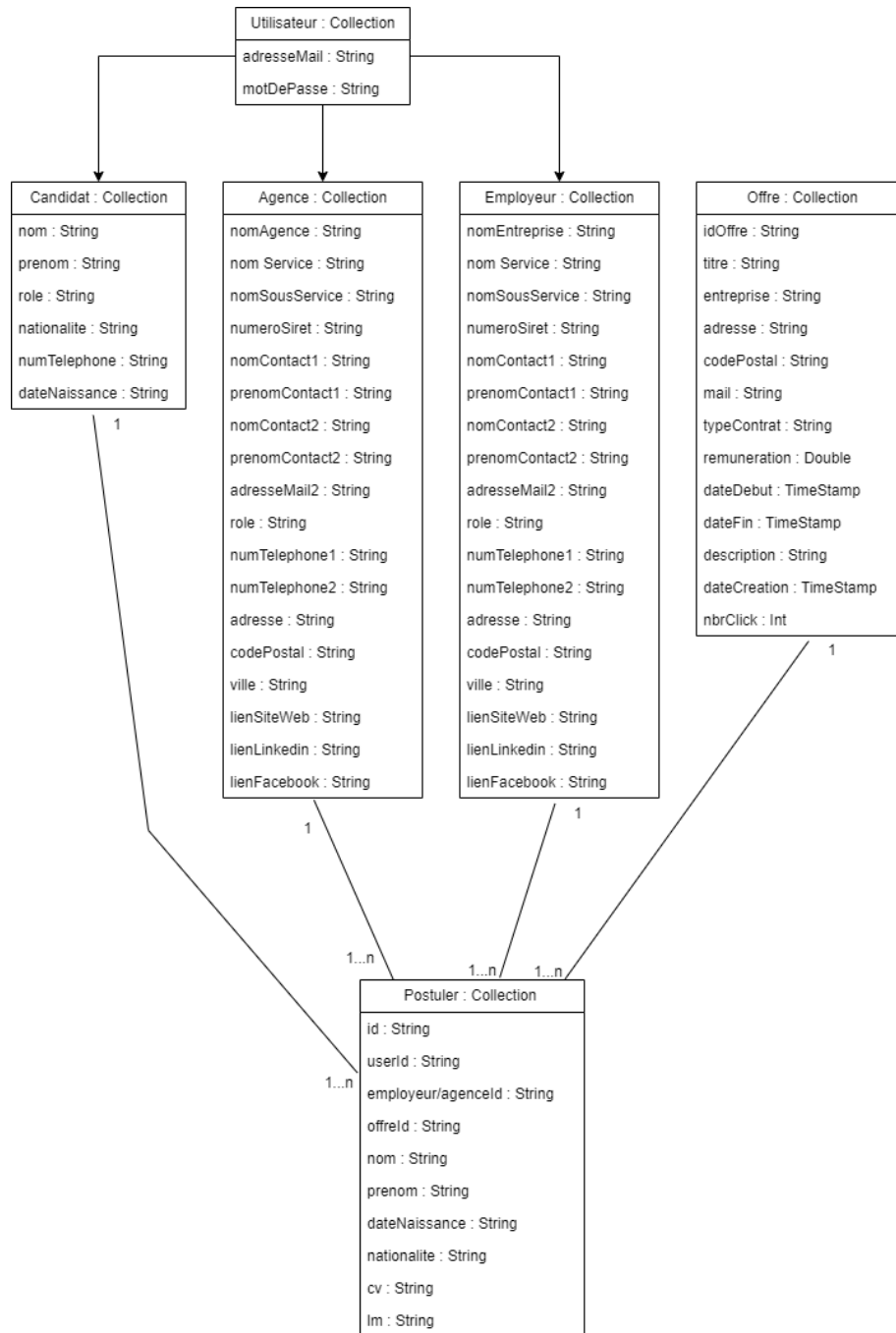


FIGURE 1.1 – Modèle relationnel de notre base de données

Pour le développement de notre application, nous avons choisi d'utiliser **Cloud Firestore**, une base

de données NoSQL orientée document de Firebase. Par rapport à la base de données en temps réel de Firebase (RealTime Database), Cloud Firestore présente plusieurs avantages :

- Une facilité de mise en œuvre des requêtes complexes
- Une persistance des données hors ligne
- Une mise à jour automatique des données en temps réel

1.1.2.2 Méthodes utilisées pour travailler ensemble dessus

Un défi auquel nous avons été confrontés dans le développement de notre application était la coordination du travail de plusieurs développeurs sur la même base de données. Pour résoudre ce problème, nous avons mis en place un système de **versionnement des donnée** qui permet de travailler simultanément sur la base de données sans interférer les uns avec les autres.

Chaque contrôleur dans notre application contient une méthode **checkAndUpdateData()** qui vérifie la version actuelle des données dans Firestore. Si la version actuelle est inférieure à la version dans notre code, la méthode remplace alors les données dans Firestore par les données dans notre code et met à jour la version dans Firestore. Cela ressemble à la commande **make :migration** du framework Symfony qui gère les migrations de base de données. Nous nous en sommes inspiré après l'avoir étudié le module HAI806I *Architectures avancées du web*.

```
fun checkAndUpdateData() {
    versionDocument.get().addOnSuccessListener { document ->
        val currentVersion = document.getLong("version")?.toInt() ?: 0
        if (currentVersion < InitialData.VERSION) {
            // La version enregistrée dans Firebase est plus ancienne que la version actuelle,
            // donc nous devons mettre à jour les données et la version.
            for (offre in InitialData.offres) {
                insertOffre(offre)
            }
            versionDocument.set(mapOf("version" to InitialData.VERSION)) // mettre à jour la
                version
        }
    }.addOnFailureListener { exception ->
        Log.e("OffreController", "Erreur lors de la récupération de la version", exception)
    }
}
```

Lors de l'exécution de l'application, nous appelons la méthode `checkAndUpdateData()` pour chaque contrôleur dans `MainActivity`. Cela garantit que les données sont à jour avant de lancer l'application. Ainsi, nous étions en mesure de collaborer efficacement tous ensemble, en travaillant simultanément sur la même base de données, sans nécessairement partager le même code au même moment.

1.1.3 Utilisation d'un tableau Trello pour organiser notre travail

Pour l'organisation du travail, nous avons pris la décision, en cours de projet, de mettre en place un tableau Trello. Cette plateforme nous a permis de répartir les tâches entre les membres du groupe, tout en classant les fonctionnalités restantes selon leur importance, de la plus critique pour le bon fonctionnement du projet à la moins significative.

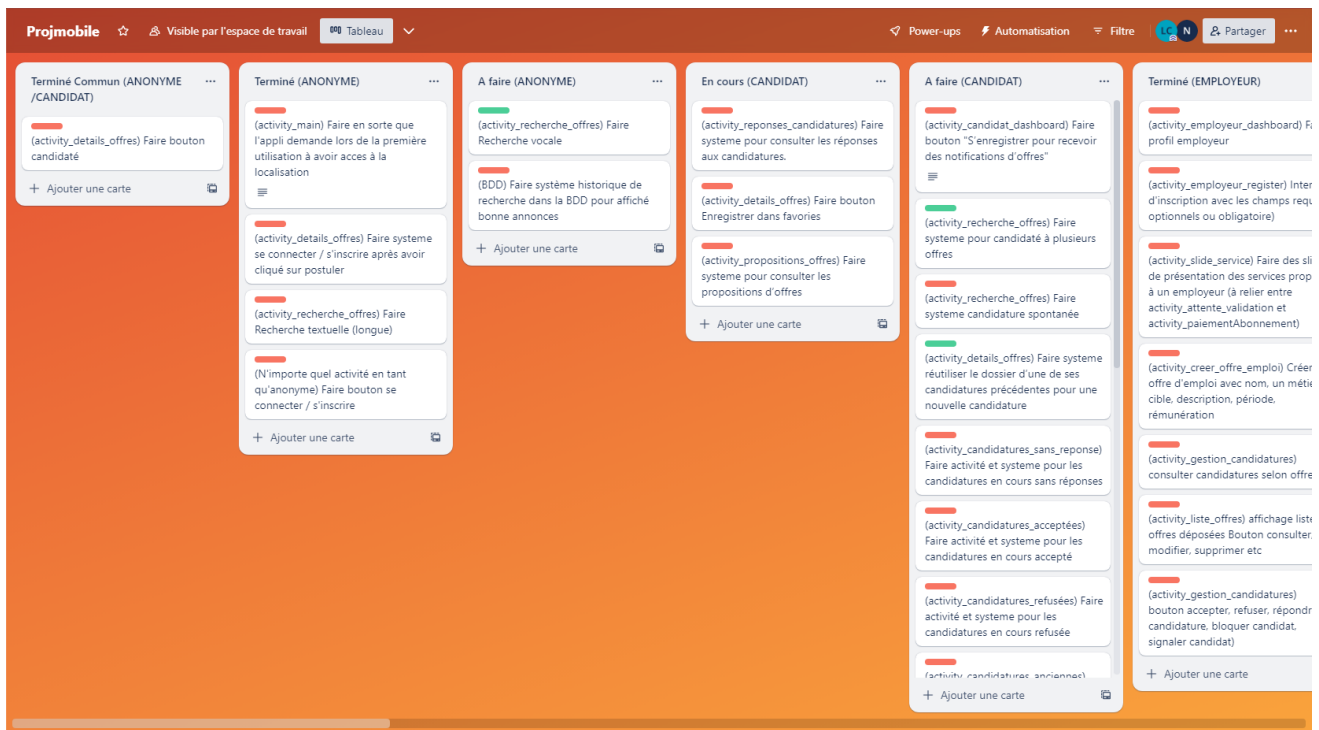


FIGURE 1.2 – Notre tableau Trello

1.1.4 Outils utilisés

Au cours de ce projet, nous avons fait usage de divers outils de collaboration qui nous ont aidé à organiser notre travail, à communiquer efficacement et à partager nos ressources. Voici la liste de ces outils :

- **Discord** En ce qui concerne la communication entre les membres de l'équipe, Discord a joué un rôle majeur. Ce dernier nous a servi à communiquer entre nous sur les différents tâches et communiquer nos avancées personnelles. Egalement il nous a servi por partager certains documents et fichiers. De plus, Discord a servi de plate-forme pour toutes nos réunions en vocal, facilitant ainsi nos interactions et notre organisation.
- **Github** Pour le partage, la gestion et le suivi des versions de notre code d'application Android, nous avons utilisé GitHub. C'est un outil familier pour tous les membres de l'équipe et il a rendu la collaboration sur le code beaucoup plus fluide et plus efficace. ²
- **Overleaf** De plus, nous avons choisi d'utiliser \LaTeX au sein d'un projet Overleaf commun au groupe pour la rédaction des différents rapports intermédiaires ainsi que de ce rapport final.
- **Android Studio** Pour la partie développement Android (applications mobiles et testing), nous avons utilisé Android Studio comme environnement de développement intégré. Le code a quant à lui été implémenté en Kotlin, langage pour lequel Android Studio offre un excellent support.
- **Firebase** Firebase est un ensemble d'outils pour l'hébergement et le développement d'applications mobiles et web, proposé par Google.
Son intégration à notre projet a été précieuse, car elle a simplifié de nombreuses tâches complexes liées au développement de l'application.
L'un des plus grands avantages de Firebase est son service de base de données en temps réel et son stockage de fichiers. Ces fonctionnalités nous ont permis de stocker et de synchroniser les données entre les utilisateurs en temps réel. De plus, grâce à la gestion du stockage des fichiers, nous avons pu facilement gérer l'upload et le téléchargement des fichiers liés à l'application, tels que les documents joints.

2.1 Liste détaillée des fonctionnalités réalisées

2.1.1 Utilisateurs non connectés (anonymes)

Cette partie du rapport traite des activités disponibles pour les **utilisateurs anonymes**.

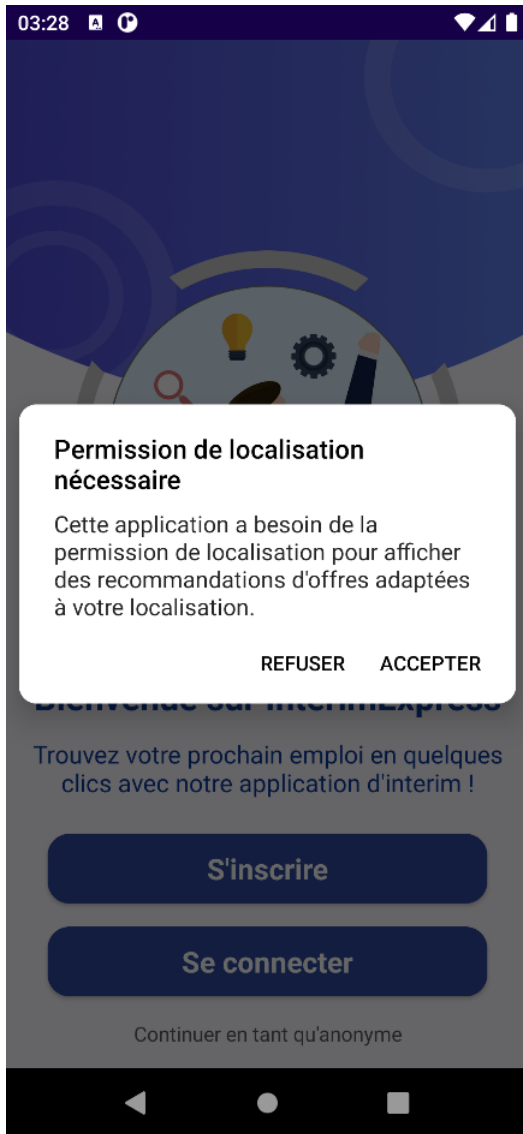


FIGURE 2.1 – Ecran de d'accueil de l'application.

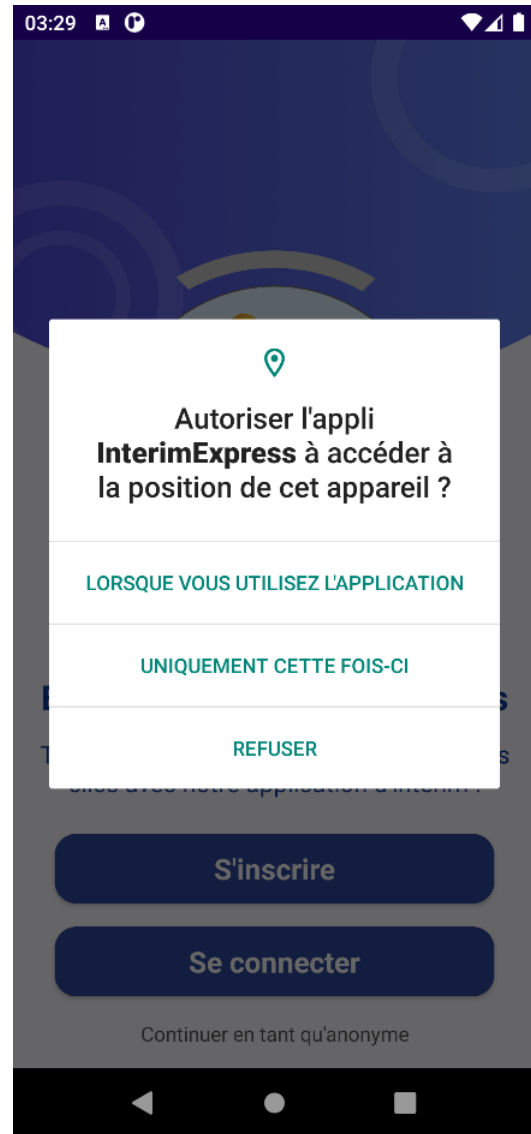


FIGURE 2.2 – L'utilisateur fait son choix.

Note:-

L'application débute en exposant la raison pour laquelle elle demande l'accès aux permissions de localisation, à savoir la possibilité d'offrir des recommandations d'offres adaptées. Ensuite, l'utilisateur est invité à accepter ou à refuser.



FIGURE 2.3 – Fenêtre d’accueil de l’application présentée à tous types d’utilisateurs



FIGURE 2.4 – Fenêtre des offres par défaut après clic sur "Continuer en tant qu'anonyme" Figure 2.3

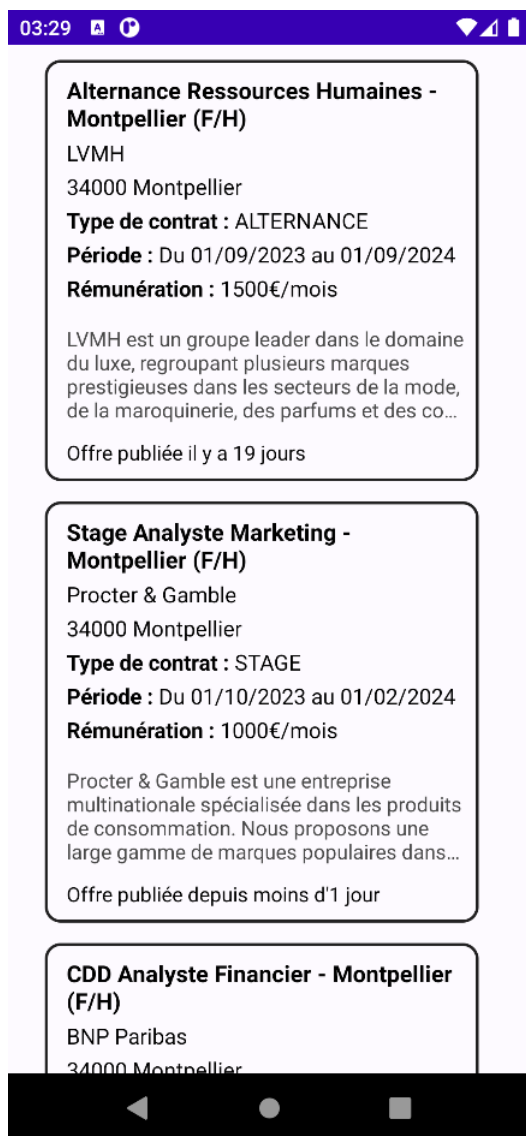


FIGURE 2.5 – Suite Figure 2.4 pour les offres affichées pour un anonyme

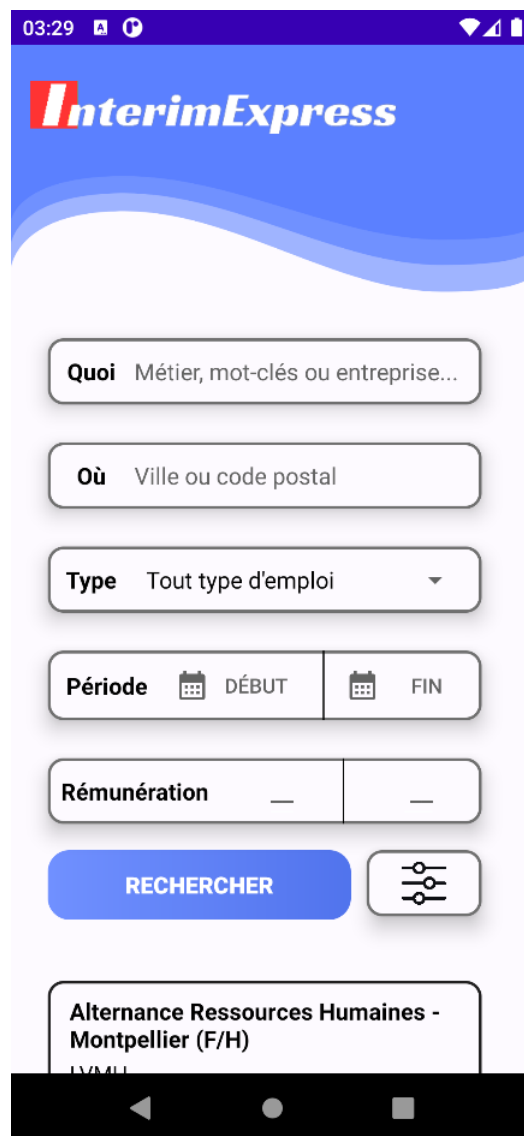


FIGURE 2.6 – Détail des options filtres de recherche possibles pour une offre

Note:-

Dans la Figure 2.5, une fois que l'utilisateur a accordé à l'application l'accès aux permissions de localisation, un algorithme détecte sa localisation à Montpellier. En conséquence, l'utilisateur ne reçoit que des offres provenant de Montpellier. Si l'utilisateur avait choisi de refuser la localisation, l'algorithme aurait alors présenté un mélange d'offres basé à la fois sur le nombre de visites (nous avons mis en place un compteur de clics pour chaque offre) et sur les offres les plus récentes.

03:31

Quoi Full Stack

Où Montpellier

Type Tout type d'emploi

Période 1/4/2023 FIN

Rémunération 2000

RECHERCHER

Alternance Développeur Full Stack - Montpellier (F/H)
 Google
 34000 Montpellier
Type de contrat : ALTERNANCE
Période : Du 01/09/2023 au 01/09/2024
Rémunération : 2500€/mois

Google est une entreprise technologique leader dans le domaine de l'Internet et des services en ligne. Nous sommes dédiés à l'innovation et à la création de produits et ...

FIGURE 2.7 – Ici ,une offre qui satisfait tous les critères de recherche a été trouvée.

03:32

InterimExpress

Quoi Full Stack

Où Montpellier

Type Tout type d'emploi

Période 1/4/2023 FIN

Rémunération 2501

RECHERCHER

FIGURE 2.8 – En modifiant la valeur de la rémunération, l'offre n'apparaît plus.

Note:-

On peut souligner la robustesse des champs de saisie, certains étant vides tandis que d'autres sont remplis, etc



FIGURE 2.9 – Après avoir cliqué sur une offre, nous obtenons sa description complète.

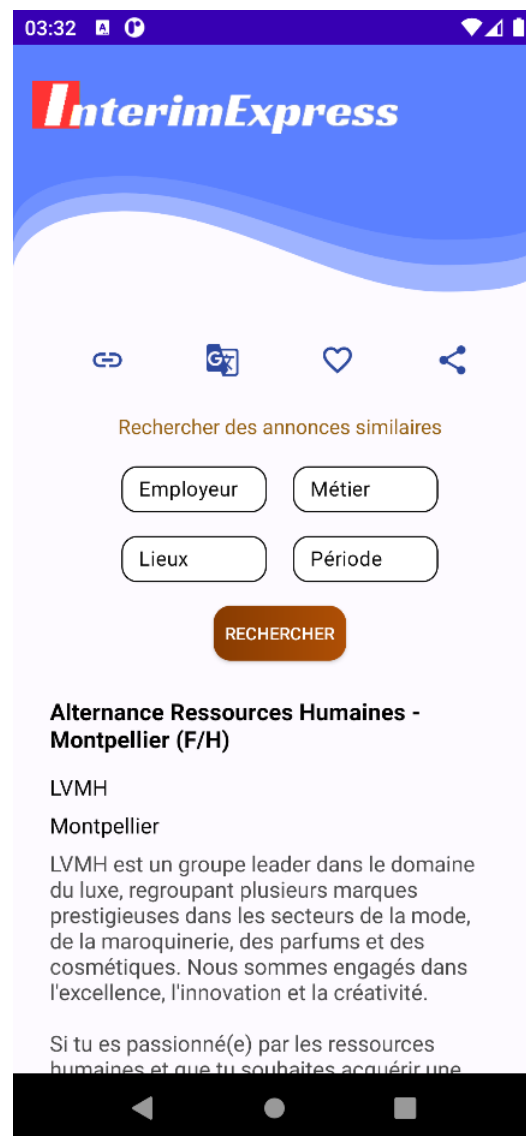


FIGURE 2.10 – Bouton "Lien vers le site web de l'annonce", "Traduire la page", "Ajouter aux favoris", "Partager", "Rechercher des annonces similaires".

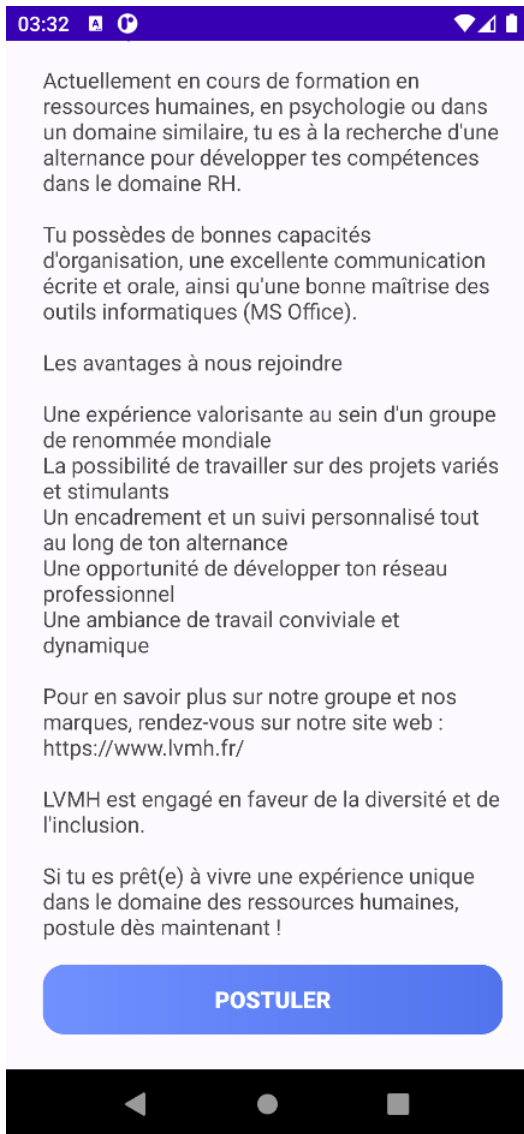


FIGURE 2.11 – Bouton "Postuler" en fin de page.

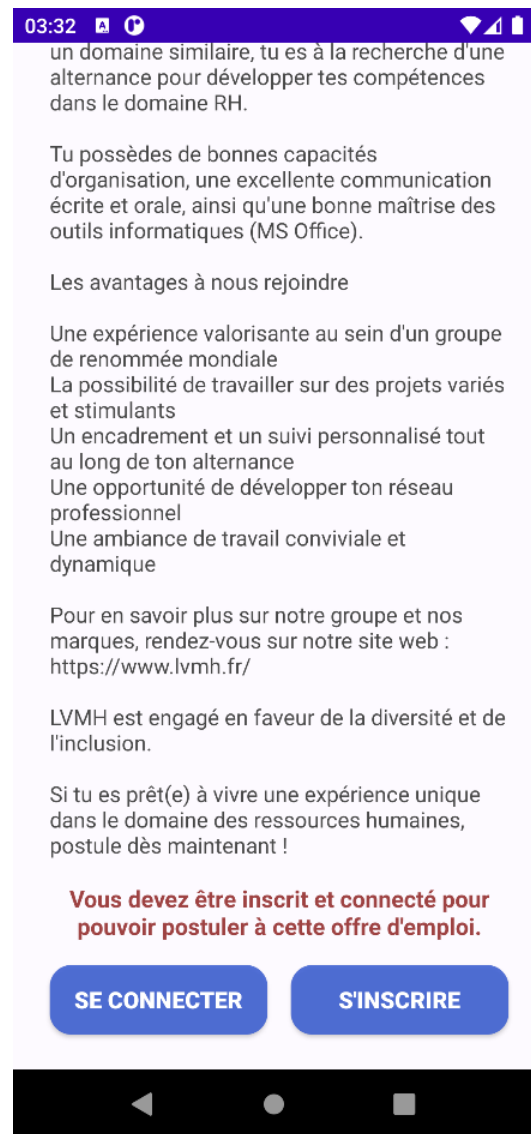


FIGURE 2.12 – Comme l'utilisateur n'est pas connecté, le message suivant apparaît après avoir cliqué sur Postuler.

2.1.2 Utilisateurs connectés en tant que Candidat

Cette partie du rapport traite des activités disponibles pour les utilisateurs connectés en tant que **Candidat**.



FIGURE 2.13 – Une fois que l'utilisateur a cliqué sur "Inscription", il sélectionne le rôle dans lequel il souhaite s'inscrire.



FIGURE 2.14 – Si celui-ci a cliqué sur "Candidat", le formulaire suivant apparaît.

03:48

Adresse mail

Votre adresse mail

Mot de passe

Votre mot de passe

Confirmation du mot de passe

Confirmer votre mot de...

Nationalité

Afghanistan

Numéro de téléphone

+33

Date de naissance

JJ/MM/AAAA

S'INSCRIRE

FIGURE 2.15 – Suite figure 2.14 Formulaire d'inscription "Candidat"

03:50

CANDIDAT TEST EMPLOYEUR TEST

AGENCE TEST

Connectez-vous à votre compte

Votre adresse mail

Votre mot de passe

Se connecter

Mot de passe oublié

FIGURE 2.16 – Interface de connexion à un compte

Note:-

En haut à gauche, nous observons des boutons qui ont été utilisés pour le testing et le débogage. Ces boutons permettent de remplir automatiquement les champs de saisie pour nous faire gagner du temps.



FIGURE 2.17 – Page d'accueil de l'application.

Note:-

Une fois que l'utilisateur est connecté, il dispose désormais d'un footer qui lui permet de revenir à la page d'accueil (celle-ci), d'accéder à son profil, ainsi que de recevoir des notifications lorsqu'il reçoit une réponse à l'une de ses candidatures. En plus de cela, il dispose également d'un avatar situé en haut, et s'il clique dessus, il sera également redirigé vers son profil.

03:51

InterimExpress

RÉUTILISER UNE ANCIENNE CANDIDATURE

Nom John **Prénom** Doe

Date de naissance 01/01/1990

Nationalité France

Curriculum Vitae

CV

Lettre de motivation *

FIGURE 2.18 – Formulaire pour postuler à une offre pour un candidat

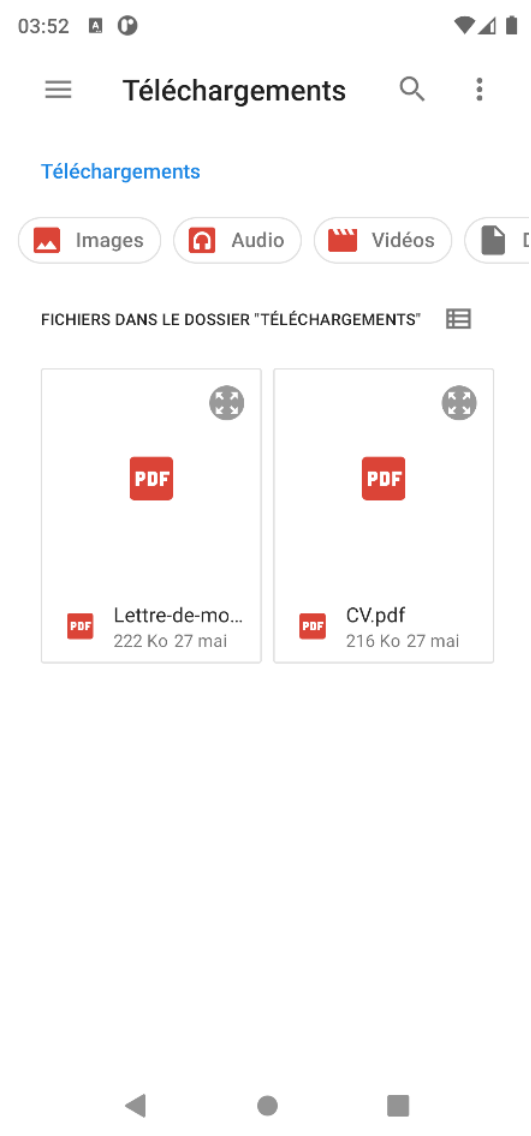


FIGURE 2.19 – Accès à l'espace de stockage pour déposer CV et Lettres

Note:-

Maintenant que l'utilisateur est connecté, il peut candidater à un emploi en cliquant sur Postuler. Il accède alors à la figure 2.18. Comme demandé dans le cahier des charges, nous avons également intégré la fonctionnalité permettant à l'utilisateur de télécharger son CV et sa lettre de motivation à partir de ses documents, afin de les envoyer à la base de données. Lorsque l'employeur recevra la candidature, il aura ainsi accès au CV et à la lettre de motivation de l'utilisateur.

03:52

RÉUTILISER UNE ANCIENNE CANDIDATURE

Nom John Prénom Doe

Date de naissance 01/01/1990

Nationalité France

Curriculum Vitae

CV Importé

Lettre de motivation *

LM

ENVOYER MA CANDIDATURE

FIGURE 2.20 – Ici, le CV à bien été importé.

03:53

John Doe
john.doe@example.com

Rechercher une offre d'emploi

FIGURE 2.21 – Profil de l'utilisateur.

Note:-

La Figure 2.21 illustre l'écran qui s'affiche lorsque l'utilisateur clique sur son avatar en haut de la page ou sur l'icône d'avatar dans le footer. En cliquant sur le bouton "Rechercher une offre d'emploi", il est redirigé vers la page d'accueil de recherche d'offres. S'il choisit de se déconnecter en cliquant sur le bouton en haut à droite, il sera ramené à la page de connexion/inscription.

2.1.3 Utilisateurs connectés en tant qu'Employeur/Agence

Cette partie du rapport traite des activités disponibles pour les utilisateurs connectés en tant qu'**Employeur**.



04:13 TESTING

Créer mon compte

Nom entreprise

Votre nom d'entreprise

Nom service/département *

Votre nom de service/département

Nom sous-service/sous-département *

FIGURE 2.22 – Formulaire création de compte "Employeur"



04:13

Nom sous-service/sous-département *

Votre nom de sous-service/sous-d

Numéro SIRET

Contact *

Nom * **Prénom ***

Son nom Son prénom

AJOUTER UN AUTRE CONTACT

Adresse mail principale

Votre adresse mail principale

AJOUTER UNE ADRESSE MAIL SECONDAIRE

Mot de passe

Votre mot de passe

Confirmation du mot de passe

Confirmer votre mot de...

FIGURE 2.23 – Suite du formulaire de création de compte

FIGURE 2.24 – Suite du formulaire de création de compte

FIGURE 2.25 – Suite du formulaire de création de compte

Note:-

Après avoir cliqué sur le bouton "Ajouter un autre candidat" de la Figure 2.23, un nouveau champ de saisie est apparu dans la Figure 2.24.

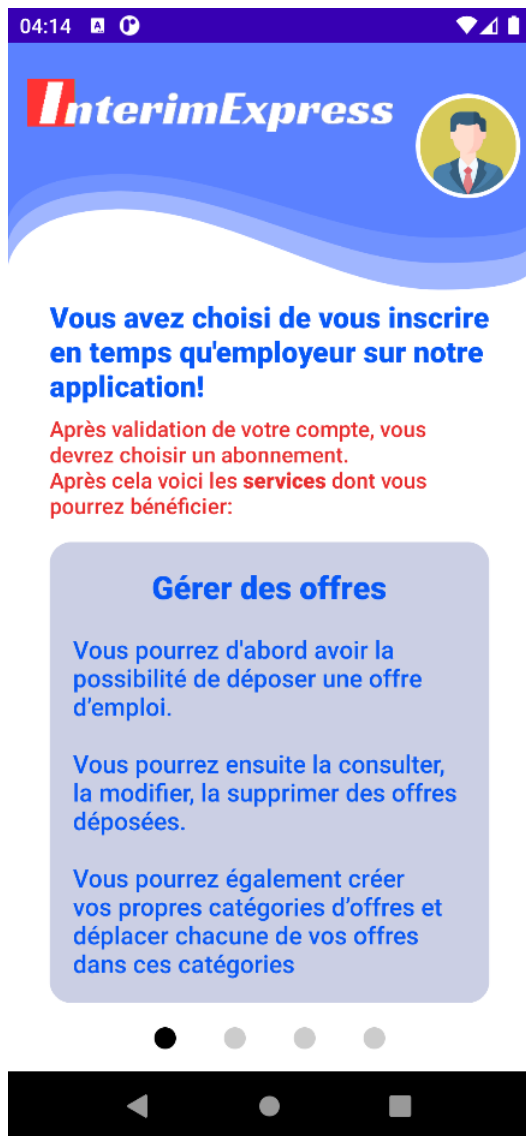


FIGURE 2.26 – Fenêtre contenant des slides présentant les différents services disponibles pour un Employeur

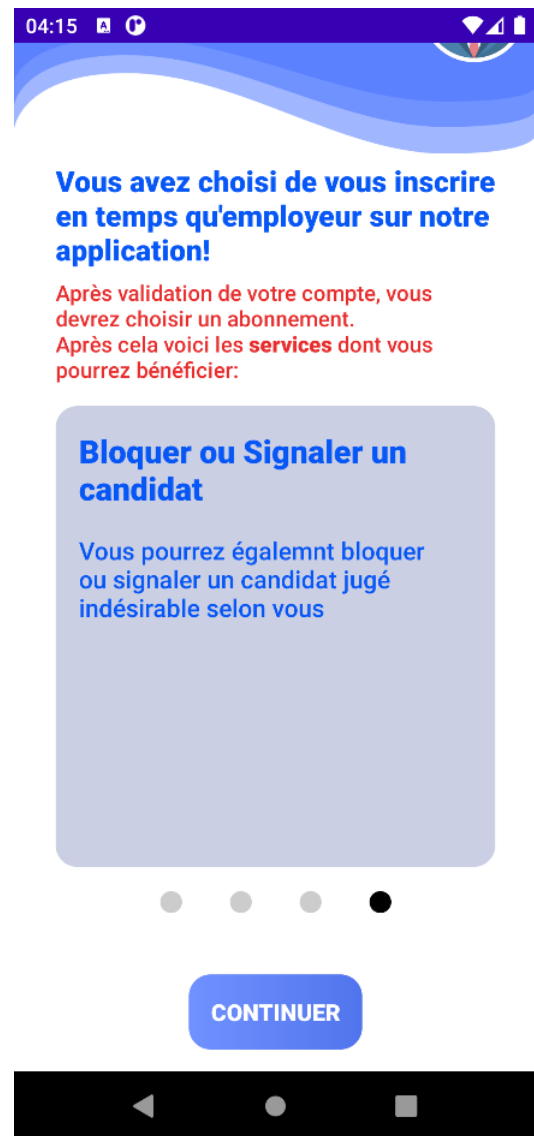


FIGURE 2.27 – Suite des slides. Le bouton "Continuer" renvoie sur le profil figure suivante

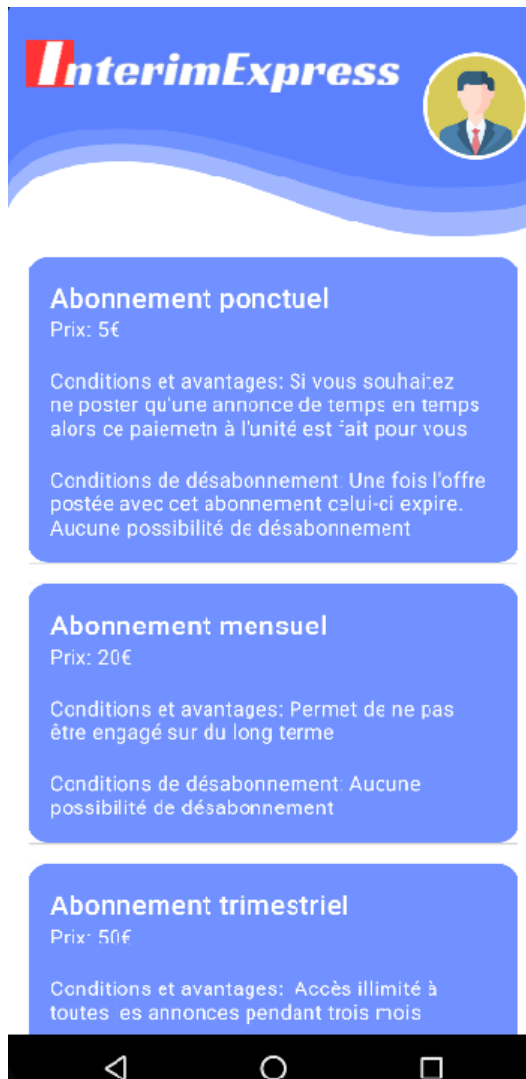


FIGURE 2.28 – Fenêtre des différents abonnements disponibles pour un employeur

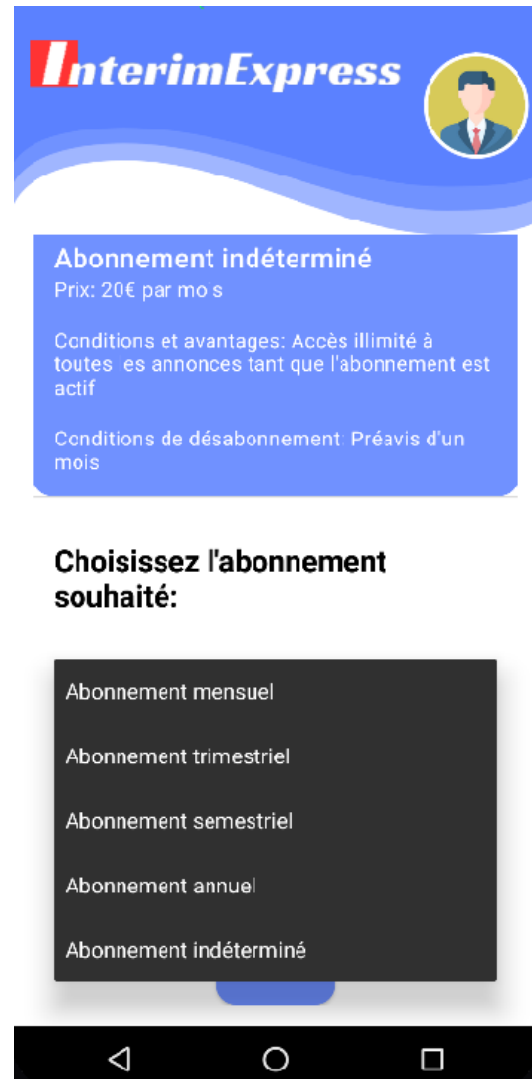




FIGURE 2.29 – Sélection de l'abonnement choisi parmi la liste des abonnements affichés

Abonnement indéterminé
 Prix: 20€ par mois

Conditions et avantages: Accès illimité à toutes les annonces tant que l'abonnement est actif

Conditions de désabonnement: Préavis d'un mois

Choisissez l'abonnement souhaité:

Abonnement annuel

Choisissez le mode de paiement souhaité:

☒ Carte Bancaire ☐ Prélèvement Automatique

PAYER

FIGURE 2.30 – Choix de l'abonnement et choix du mode de paiement




Abonnement choisi:
 Abonnement annuel

Prix abonnement:
 160€

Mode de paiement:
 Carte Bancaire

Saisir coordonnées bancaires

1234 1234 1234 1234 MM/YY

CONFIRMER ET PAYER

FIGURE 2.31 – Récapitulatif de l'abonnement et mode de paiement choisi + saisie seulement des coordonnées bancaires



FIGURE 2.32 – Profil de l'utilisateur avec les différents fonctionnalités disponibles

FIGURE 2.33 – Formulaire de création d'une offre d'emploi après avoir cliqué sur "Créer offre emploi" sur le profil

Créer une offre d'emploi

Nom offre
Caissier Carrefour (H/F)

Nom entreprise
Carrefour

Type de contrat
Type CDD

Rémunération
1350

Période
1/4/2023 31/7/2023

Description offre
Carrefour est une entreprise leader dans la grande distribution...
Nous recherchons un profil (nous acceptons les candidatures sans aucune expérience)...
Dans ta mission ,tu devras...

FIGURE 2.34 – Remplissage du formulaire et des différents champs requis

Créer une offre d'emploi

Consulter mes offres

Gérer les candidatures

Candidatures acceptées

FIGURE 2.35 – Redirection sur le profil après avoir validé la création de l'offre

Note:-

Dans la Figure 2.31, nous remarquons que la cloche de notification dans le footer ne présente aucune notification pour le moment, ce qui est normal puisque l'offre vient d'être créée. Nous constaterons un peu plus bas que la couleur de la cloche changera lorsque nous recevrons une candidature.

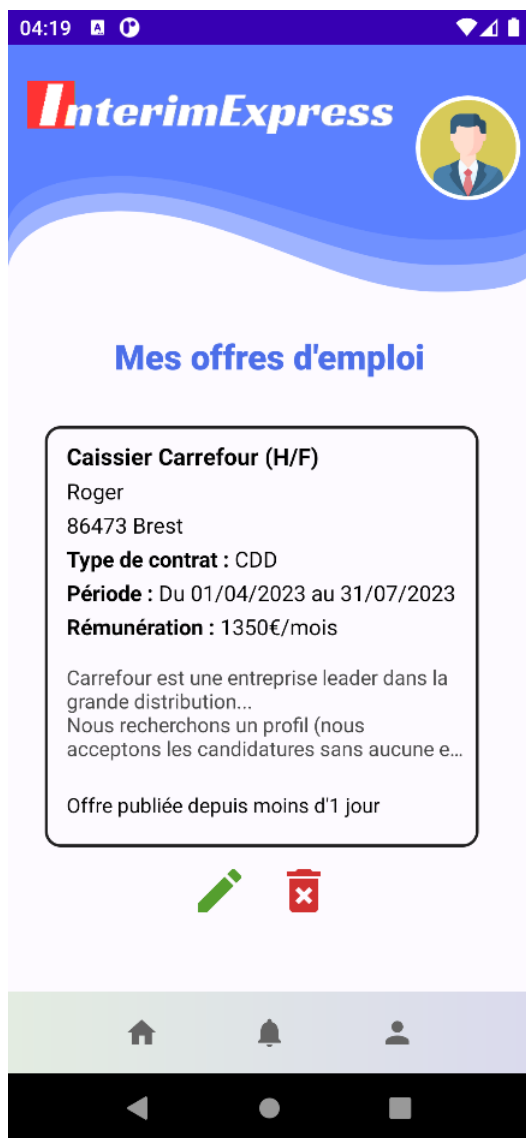


FIGURE 2.36 – Après avoir cliqué sur le bouton "Consulter Offres" dans le profil, une fenêtre de consultation des offres que l'on a créer s'affiche, offrant la possibilité de les modifier ou de les supprimer.

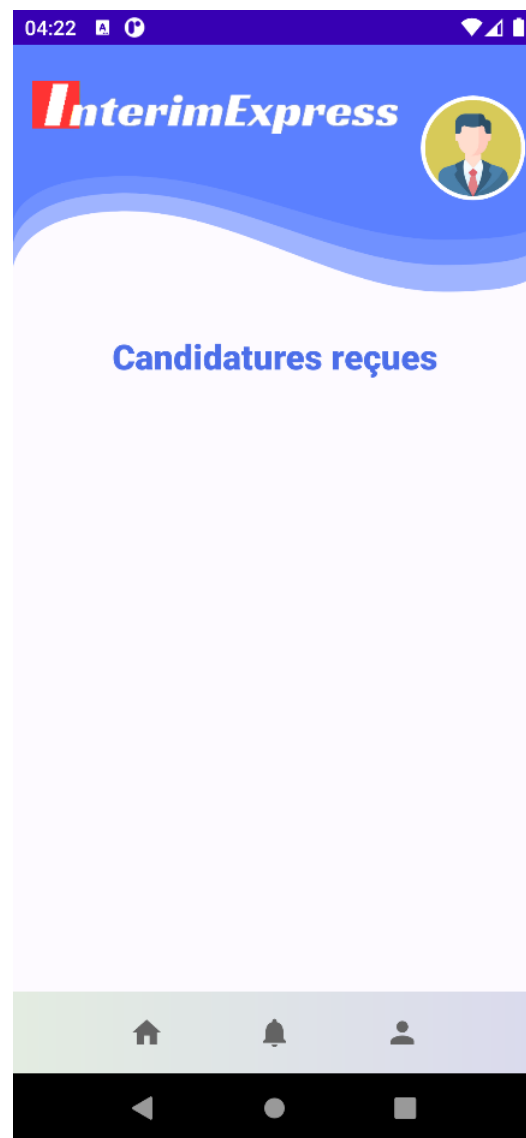


FIGURE 2.37 – Fenêtre de consultation des candidatures (vides pour le moment) reçues après avoir cliqué sur "Gérer les candidatures" dans le profil

Note:-

On retrouve ici l'offre d'emploi appelée **Caissier Carrefour (H/F)** créée plus haut par l'employeur dans le formulaire. Naturellement, personne n'a encore postulé à notre unique offre que nous venons de créer, donc la page "Candidatures reçues" est vide pour le moment.



FIGURE 2.38 – L’offre que nous avons créer en tant qu’Employeur apparaît bien pour les Candidats.



FIGURE 2.39 – Apparition d’une notification en bas du profil dans le footer.

Note:-

Nous nous sommes connectés avec le compte d’un candidat quelconque dans la figure 2.34 afin de bien prouver que l’offre apparaît. Ensuite, nous avons postulé à cette offre avec ce compte (nous y avons déposé un CV et une lettre de motivation).

Dans la figure 2.35, nous nous reconnectons sur notre compte Employeur, cette fois-ci nous voyons une notification qui apparaît ! C’est pour signifier que nous avons bien reçu une candidature.

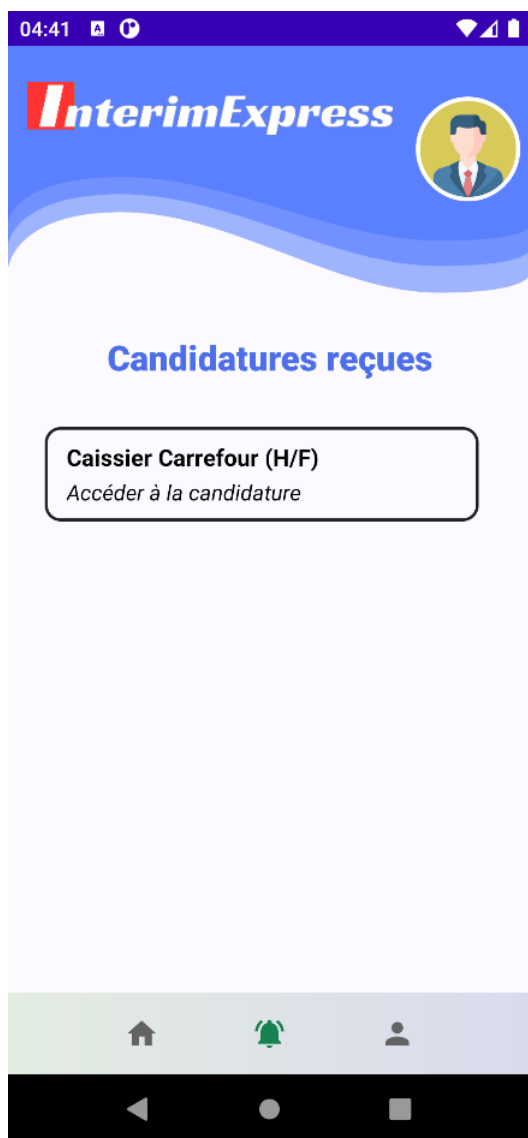


FIGURE 2.40 – Apparition cette fois de la candidature reçue dans la page "Gérer les candidatures"

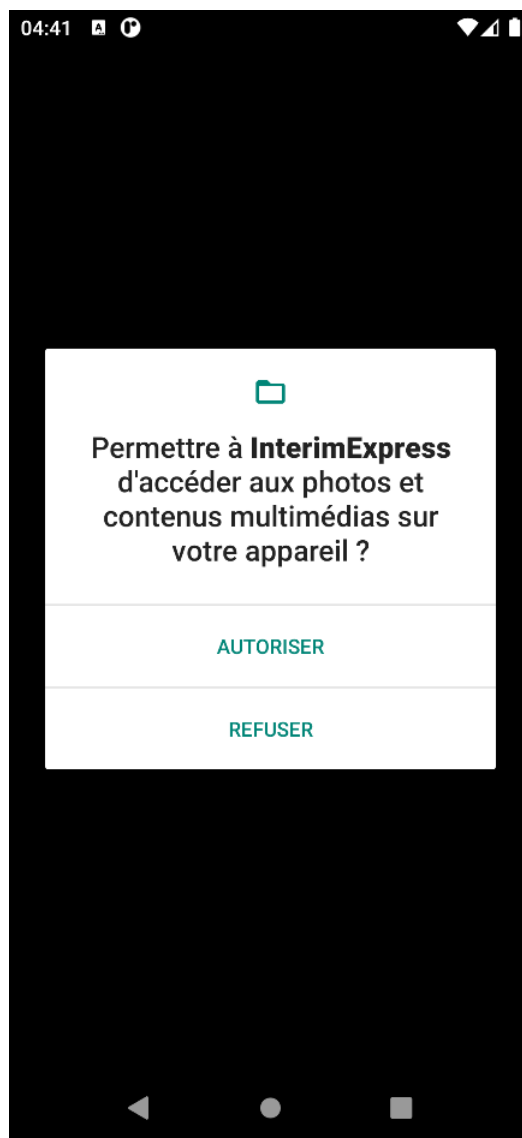


FIGURE 2.41 – Après avoir cliqué sur la candidature ,l'application demande l'accès à l'espace de stockage pour permettre à l'employeur de télécharger les documents pdf fournis par le candidat.



FIGURE 2.42 – Affichage de la candidature reçue avec possibilité d'accepter ou de refuser celle-ci.

Note:-

Si nous cliquons pour télécharger le CV ou la LM, ils apparaissent dans l'espace de stockage du téléphone. L'employeur peut alors prendre le temps de les lire avant de faire son choix pour accepter ou refuser la candidature.

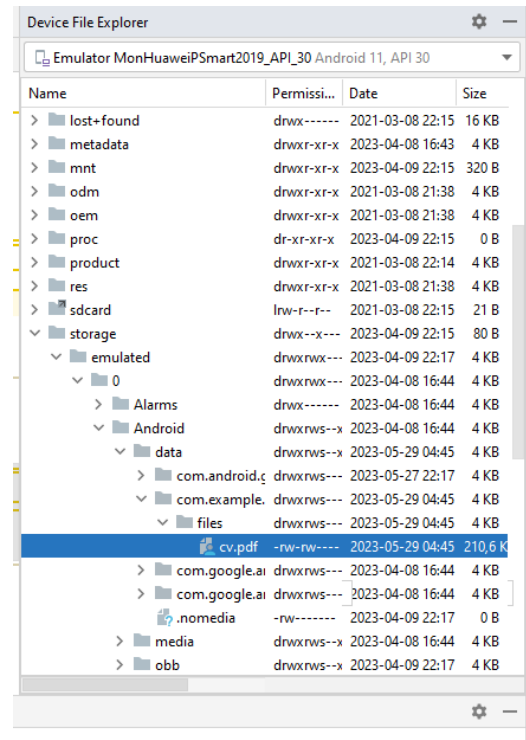


FIGURE 2.43 – Apparition du CV téléchargé dans l'espace de stockage du téléphone. (Emulateur android)

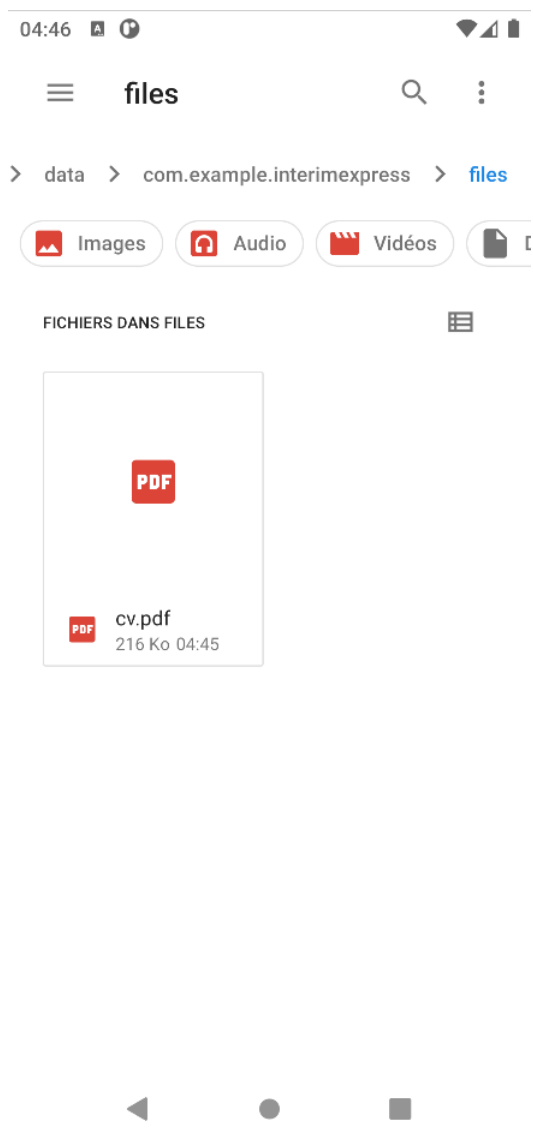


FIGURE 2.44 – Nous retrouvons bien le pdf téléchargé dans l'espace de Stoackage.

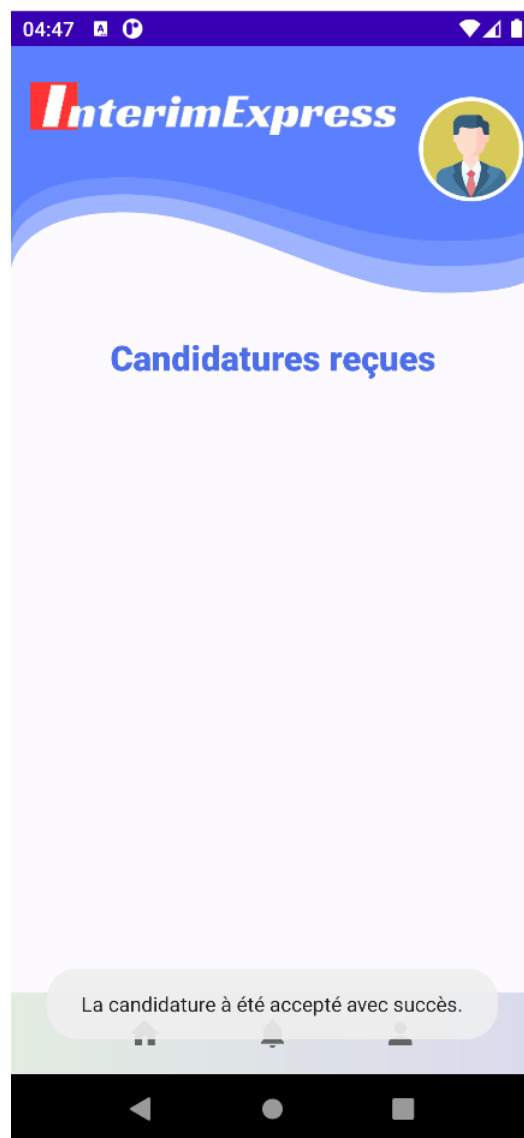


FIGURE 2.45 – Nous avons cliqué sur "Accepter", un message Toast apparait en bas et la notification du footer disparaît de notre profil.



FIGURE 2.46 – Le candidat est notifié.

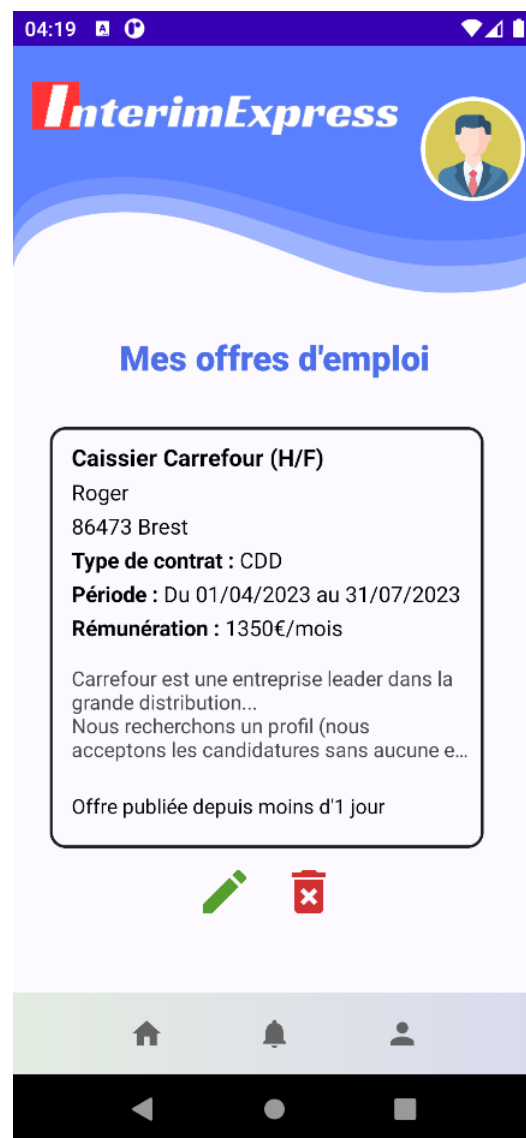


FIGURE 2.47 – Bouton Edit et Supprimé une offre.

Note:-

Nous nous reconnectons maintenant avec le compte du candidat qui a postulé. Ce dernier reçoit maintenant à son tour une notification dans son dossier qui lui indique que sa candidature a été acceptée par l'employeur.

Revenons à notre employeur, la figure 2.43 et les suivantes montrent le fonctionnement des boutons Edit et Supprimé

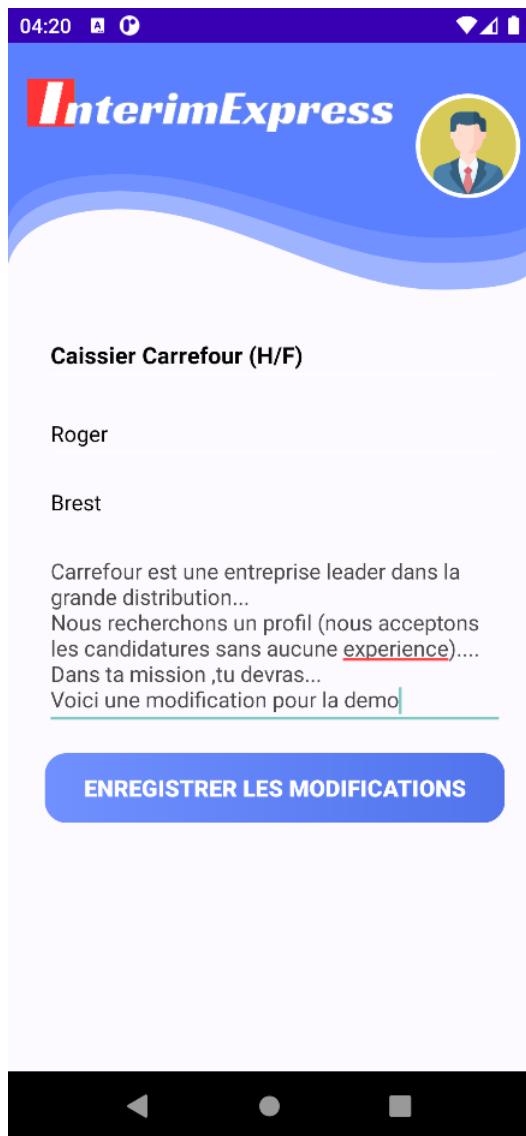


FIGURE 2.48 – Si nous cliquons sur "Modifier", nous pouvons modifier l'offre. Les nouvelles données seront bien prises en compte par la base de données.



FIGURE 2.49 – Si nous cliquons sur "Supprimer", nous recevons une alerte.



FIGURE 2.50 – Après avoir cliqué sur "Oui", l'offre est bien supprimée. Elle n'apparaît plus dans la base de données et donc dans notre profil également.

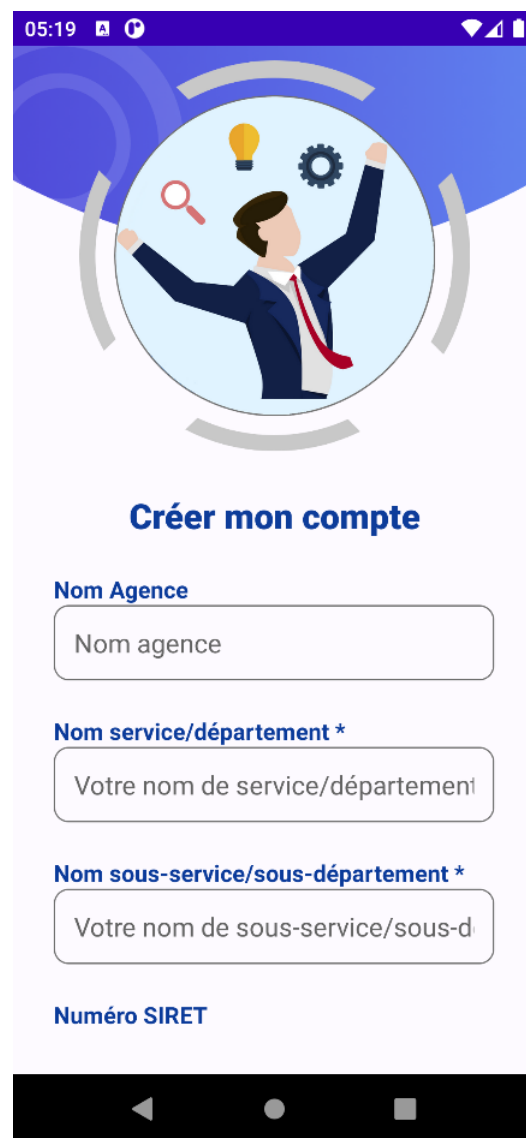


FIGURE 2.51 – Formulaire de création d'un compte Agence.

Note:-

Enfin, nous voulions vous montrer que les agences ont également été implémentées. Elles possèdent exactement le même fonctionnement que les employeurs. Voici, par exemple, le formulaire de création d'un compte agence.

3.1 Conclusion

Au cours de ce projet mobile, nous avons réussi à mettre en œuvre plusieurs fonctionnalités essentielles. Nous nous sommes premièrement focalisés sur les fonctionnalités essentielles telles que la gestion des inscriptions, l'élaboration d'offres pour les employeurs et les agences, et la capacité pour un candidat de postuler à une offre.

Pour cela, de nombreuses activités ont pu être implémentées pour les divers rendus d'écrans et l'implémentation des logiques correspondantes à celle-ci a également été faite. Chaque fonctionnalité constituait une logique assez différente ce qui nous a permis de développer plusieurs points différents.

Cependant, nous aurions avec davantage de temps pu implémenter davantage de fonctionnalités.

Pour terminer, ce projet a été l'occasion pour nous de nous familiariser avec des outils tels que Firebase, et de comprendre comment gérer la logique d'une application mobile sur un projet. L'utilisation de ces outils a non seulement renforcé nos compétences techniques, mais aussi approfondi notre compréhension de l'architecture et de la logique des applications.